Noise-agnostic Adaptive Image Filtering without Training References on an Evolvable Hardware Platform

Javier Mora, Ángel Gallego, Andrés Otero, Eduardo de la Torre, Teresa Riesgo
Centro de Electrónica Industrial
Universidad Politécnica de Madrid
Madrid, Spain
{javier.morad, angel.gallegog, joseandres.otero, eduardo.delatorre, teresa.riesgo}@upm.es

***Abstract*— One of the main concerns of evolvable and adaptive systems is the need of a training mechanism, which is normally done by using a training reference and a test input. The fitness function to be optimized during the evolution (training) phase is obtained by comparing the output of the candidate systems against the reference. The adaptivity that this type of systems may provide by re-evolving during operation is especially important for applications with runtime variable conditions. However, fully automated self-adaptivity poses additional problems. For instance, in some cases, it is not possible to have such reference, because the changes in the environment conditions are unknown, so it becomes difficult to autonomously identify which problem requires to be solved, and hence, what conditions should be representative for an adequate re-evolution. In this paper, a solution to solve this dependency is presented and analyzed. The system consists of an image filter application mapped on an evolvable hardware platform, able to evolve using two consecutive frames from a camera as both test and reference images. The system is entirely mapped in an FPGA, and native dynamic and partial reconfiguration is used for evolution. It is also shown that using such images, both of them being noisy, as input and reference images in the evolution phase of the system is equivalent or even better than evolving the filter with offline images. The combination of both techniques results in the completely autonomous, noise type/level agnostic filtering system without reference image requirement described along the paper.**

Keywords*— *evolvable hardware; evolutionary algorithms; adaptive systems; image filter; reference image; camera

I. INTRODUCTION

Self-adaptive systems are a widely used approach for autonomous system design that allows them to automatically adapt to the conditions of a certain problem without requiring user control. This is usually achieved by feeding the system with known test data and comparing the output with a golden reference. If output and reference differ too much, this means that the system needs to be modified, because the bigger the difference, the further the system is from the ideal solution.

One way to do this is by using *evolutionary algorithms* (EA), which are mainly optimization methods used to find a solution based on a target criterion, typically called *fitness function*. These algorithms are inspired by natural selection and follow the premise of "survival of the fittest", seeking better solutions by applying random changes to already known ones, in an attempt to improve their worst features and hence reach optimized solutions of the problem.

Hardware architectures able to perform *dynamic and partial reconfiguration* (DPR), such as RAM-based FPGAs, may self-reconfigure themselves and take advantage of the possibility of intrinsically modify their logic, so by such evolutionary algorithms that may achieve inherent self-adaptivity. These are the basic elements to obtain Evolvable hardware (EHW) systems.

Typically, operation time of these systems is divided into two stages: first, a training stage is carried out, in order to achieve a proper configuration which will depend on the different operating conditions (environment, inputs, the system itself, presence of faults…). This stage can take several minutes, during which the system will be halted. Next, a normal or mission operation stage starts, in which the system works well in those particular conditions, according to the configuration achieved in the previous stage. In order to also have fault tolerance, these two stages could be alternated during the entire lifetime, for instance combined with a fault detection mechanism that launches the evolution phase again to overcome the permanent fault that had occurred.

However, the described mechanism has several problems in real time conditions. The main problems to consider are:

- It requires a way to identify the type and level of noise that is going to be filtered, which will be different on each situation (otherwise the filter would not be adaptive).
- Moreover, it relies on having an appropriate and accurate model for that noise (which can be generated and added online or offline), in order to create the test images to train the system from the golden reference.
- Finally, it requires a way to store, receive or generate the golden reference and the training images with the generated noise.

These problems are hard to solve without overcomplicating the system (for example, adding external storage, a communication system, a noise detector and a noise classifier, a configurable noise generator...), so finding a way to skip them would be beneficial to the system. Along this paper, a solution to all these problems is shown, by combining two

main techniques: one of the techniques is based on using two identical images with added uncorrelated noise for both the training image and the reference image. As it was shown in [1], the usage of the same image with the same type and level of noise, but with zero correlation between them, allows the system to obtain valid solutions for the configuration of the filter in the training stage.

On the other hand, the second technique is based on using two consecutive frames from a camera as the 'identical' couple of images mentioned before, which allows to get rid of the storage, reception or generation of golden reference images. Additionally, since the images used for training come from the camera, there is no need for a noise identification stage and/or an accurate model of the noise. The implementation of both techniques together is the main novel contribution of this work. The quality of the filters achieved using these techniques needs to be verified statistically. Experimental results will show that the use of these techniques combined with a proper selection of the images to be set for training provide results which are equivalent (and in some cases even better) than with a traditional evolution run with golden pre-computed reference images.

The self-adaptivity techniques that are proposed in this paper have been implemented on a system which is capable of performing intrinsic evolution by using the native dynamic and partial reconfiguration techniques of the FPGA that holds the complete system. Details on the architecture are contained in [2], but the important features are extracted and presented in section III within this paper for completeness. Systems that are able to evolve intrinsically have other advantages like the capability of self-healing themselves by launching a new evolution run which remaps the generated logic by avoiding the use of the faulty elements. In our system, an analysis of the fault tolerance is presented in [3]. Although this is not the scope of the present paper, this feature must be taken into account in order to appropriately evaluate the added benefits of both self-adaptivity and self-healing, making the system more autonomous.

The paper is structured as follows: Related work is shown in section II. The architecture of the whole system is summarized in section III. In section IV, an analysis of the main possible solutions is done, including the main aspects of the selected methods. Results obtained and the main benefits of this architecture are shown in section V. Finally, conclusions and future work are drawn in section VI.

II. Related work

Image filtering is the process by which the noise in an image is removed to have a clearer one. This noise can come from different sources: malfunctioning pixels in the camera sensor, dust in the lens, the circuitry of the camera, and mainly because of the transmission in a noisy channel [4]. The most common noise model is impulse noise, in which perturbations affect a few random pixels, leaving the rest of them unaltered. Impulse noise models are either salt and pepper noise (which means that some pixels are replaced by the highest or the lowest value of their range) or random-value noise (any value of the range). There are many different approaches to tackle this problem, most of them being nonlinear filters. The median filter has been one of the most used filters, but it presents several drawbacks and it has been substituted by more advanced ones. Even so, it is still used as a reference for comparison.

To deal with the problems of the median filter, different methods try to improve performance with a pre-analysis of the image in order for the filter to identify the noisy pixels, and hence be more efficient, trying to achieve the so-called detail preserving [5]. Among these new algorithms are Adaptive Median (AM) filters ([6]), Switching Median (SM) filters ([7]), and Weighted Order Statistics (WOS) filters ([8]). These decision-based filters take advantage of the pre-analysis and only modify the noisy pixels, leaving the rest of them unaltered, and as a result, very high quality images are obtained with really high noise rates (up to 90%, as shown in [9] and [10]). However, the complexity of these filters and the tendency to implement them in software implies high times in order to pre-process and filter the image, and so, they are not suitable for real time processing even for low frame rates.

To improve the speed, implementing these filters in hardware is another alternative. For instance, a hardware implementation of the AM filter is described in [11], which achieves throughputs up to 300 megapixels per second, making it suitable for real time applications, and is able to filter images in which up to 60% of pixels are corrupted by using a 7×7 window. Other hardware implementations of image processing filters are shown in [12] and [13], with different tradeoffs between resource usage and performance. However, all these solutions present high implementation costs, and their adaptability is based on analyzing the local properties of the signal, rather than adapting the system as a whole, designing new configurations at runtime and automatically in order for the filter to solve the given problem.

On the other hand, evolutionary techniques are a hot topic within the advanced design methodologies and the advanced processing architectures of autonomous and self-adaptive systems. In the hardware field, lots of advantages are obtained by the usage of these implementations on FPGAs, using the aforementioned EHW. There are two main manners of implementing EHW: by creating a circuit that can switch between different functionalities (for example, using a multiplexor) or by reconfiguring the FPGA fabric using DPR.

First attempts in this field using DPR were unfeasible due to the slow reconfiguration times and the obfuscated structure of the configuration bitstream, so the other approach was often preferred. This is the case of implementations such as *virtual reconfigurable circuits* (VRC) [14], which achieve high reconfiguration speeds at the expense of a high resource usage. An example application of such a system for image filtering with EHW is proposed in [15], which uses VRC together with a processing architecture known as *Cartesian genetic programming* (CGP) [16]. In [17], this solution is implemented using a hardware implementation of the EA.

However, interest in DPR has grown in the last years, and new approaches have appeared that allow for more flexibility and faster reconfiguration speeds. This manner of design allows the system to be flexible and adaptive, and it can be

applied in a wide range of applications, from image filtering tasks ([18]) to data classification systems, such as the one shown in [19].

An intermediate approach between DPR and VRC is using the *shift register LUT* (SRL) behavior of certain Xilinx FPGAs as a way to change its logic [20]. This manner of reconfiguring is much faster, but its possibilities are more limited.

III. System Description

The system has been implemented on an FPGA, following a *System on Programmable Chip* (SoPC) approach, including the main components: i) the reconfigurable region where the filter processing logic is implemented, which has been designed as an array of processing elements; ii) the evolutionary algorithm, which takes decisions on how and when to mutate the array according to the evolutionary rules, iii) the reconfiguration engine (RE) which is in charge of modifying the array functionality by managing the internal configuration port of the FPGA, with the reconfiguration commands coming from the EA, and iv) the fitness evaluation unit, which computes the accumulated sum of the differences, pixel by pixel, of the output image and the reference image, and whose output value (fitness and latency of the processing) are sent to the Microblaze so that the EA performs a new candidate selection.

The processing array is a data processing architecture known as *systolic array*, in which several processing elements (PEs) are arranged in a matrix (in this case, a 4×4 matrix). Each of these PEs performs a simple operation (addition, subtraction, average, data copying…) in a single clock cycle, using the data that may come from the processing elements situated above, to the left, both above and left, or none (there is a PE giving 0xFF value at the outputs, but it does not read data). The result is sent to both the PE below it and to its right. Fig. 1 shows a schematic diagram of a PE and the processing array structure. The evolutionary algorithm (EA) decides which PE out of a library of PEs is to be placed in each of the 16 different positions. This architecture is faster than CGP since data moves from a PE to a contiguous one rather than one placed in an arbitrary position, thus making the data paths shorter and not requiring multiplexors to select the PE inputs as the paths are prefixed.

In this particular implementation, the data to be processed will be 8 bit pixels from an input image which has some kind of noise that has to be removed. These pixels are fed to the array as a 3×3 pixel window which surrounds the pixel to be filtered. This window slides over the whole image, moving one pixel per clock cycle, until it reaches the last pixel of the image. Multiple data values are fed to the array from the top and left sides. The EA decides which of the 9 pixels of the sliding 3×3 window is selected in each of these inputs. Also, the result is extracted from one of the PEs on the right side, which is selected also by the EA. As it can be seen, this architecture is able to process one data value per clock cycle in a segmented manner, with certain data propagation latency, which depends on the selected PEs, selected pixels at the inputs and selected output.

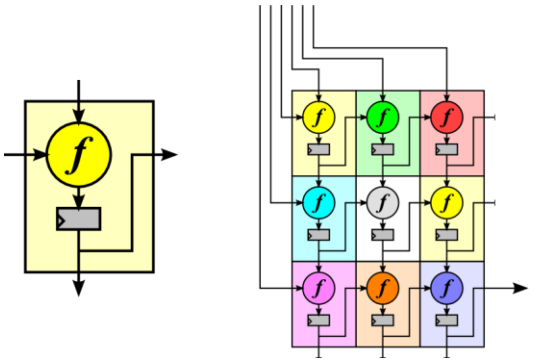

Fig. 1 Structure of a PE and architecture of the systolic array, which consists of different PEs. Each PE communicates with its neighbors, and performs one operation per clock cycle.

In order to automatically configure the filter to perform a specific task, an EA running on the MicroBlaze is used. The EA implemented here is a simple (1+λ) algorithm in which a population of λ+1 candidate filter configurations are evaluated. Each of these candidates is described as a set of parameters, each of which indicates what function each PE performs or which input or output is selected.

This algorithm starts with a population of λ+1 randomly generated candidates. Then it evaluates each of these candidates by reconfiguring the filter with the parameters from the candidate, filtering a test image, which is an image to which noise has intentionally been added, and comparing the result pixel by pixel with a reference image, which is the unaltered image, as shown in Fig. 2. The fitness function is calculated as the sum of absolute differences between pixels (SAE, sum of absolute errors), so the lower the fitness value, the better the filter. The fitness values from all candidates are compared, and the candidate with the lowest fitness is chosen as the parent for the next generation. This parent is mutated by replacing *K* randomly chosen parameters with new random values, creating a new child candidate (*K* is known as "mutation rate"). The process is repeated λ times, generating λ new candidates from the same parent, so that the population now has λ+1 members including the parent. The EA is executed for a certain number of generations, which in our case is stopped after a given number of generations have taken place. This solution will be a filter able to process other images with the same type of noise.

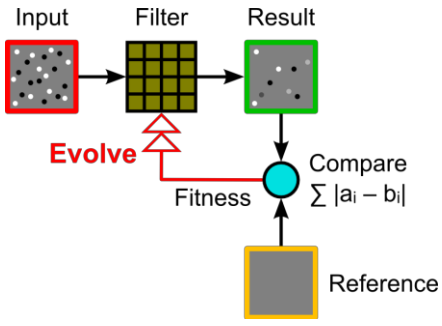

Fig. 2 Evolutionary loop scheme with the array, the test input image, the golden reference and the output filtered image. Fitness is calculated and used in the loop to improve the configuration.

The PEs which are selected by the EA are set into the array using the DPR capabilities of the FPGA. In order to do this, a RE has been developed, which will be able to modify the systolic array while the rest of the system keeps working. The RE is able to relocate the bitstream position depending on the PE selected to be changed, so the PE library requires only one copy of each PE, for a neutral position.

In addition to the aforementioned components, the SoPC has external memory controllers for a CompactFlash card and RAM memory, which will be used for loading and storing the images and the partial circuits for each PE function, and a camera controller that allows the system to take images from a CMOS sensor. This camera will allow the system to be used for real time image processing.

IV. Problem Description and Possible Solutions

The problem of obtaining the reference in autonomous systems in order to have adaptivity to different types and levels of noise is the main goal to be solved. In general, different solutions are traditionally considered, although in principle all methods are variations of the following ones:

- To generate the reference image starting from a filter model, which could possibly be in SW (but it would be slow), and apply this model to a noisy image, so that the result image is used as the reference image to follow during evolution. This solution requires the noise to be pre-characterized and identified in order to apply the appropriate filter or, if a generic noise reduction filter is used, then filter performance may be very poor.
- To use a noise generator model to generate the input image to be used during evolution, while keeping the noise-free image as the solution to evolve to. This also requires noise pre-characterization.
- To feed the system with a known test pattern. This involves the system to be able to point, during mission operation, at a specific know position, in order to compare the noisy image acquired this way with the pre-stored reference pattern. This is difficult to achieve in many cases.

As it can be seen, either noise analysis and pre-characterization, poor performance, or lack of a reference to have a kind of self-calibration are the main problems of the previous methods. So, the solution that is proposed is based on using a second noisy image as reference image.

Let's consider two images with the same type of noise, for example salt and pepper noise (in which randomly selected pixels from the image have been replaced with black or white pixels). If the noise in both images is not correlated, some pixels that are noisy in one of the images will be clean in the other one. Because of this, an evolvable filter that uses one of the noisy images as input and the other one as reference would attempt to make the noisy pixel look like the one with no noise (Fig. 3). For this reason, using two noisy images as input and reference for training can be a valid alternative to having a clean reference.

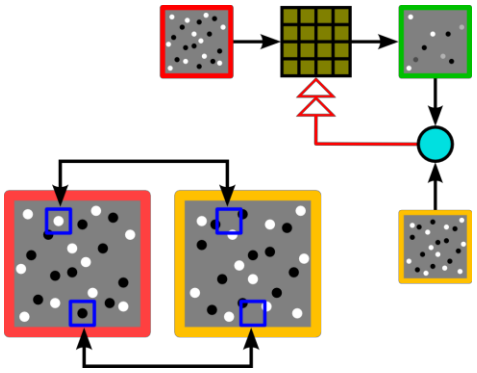

Fig. 3 Same evolutionary loop but using two non-correlated noisy images.

The requirement of non-correlated noise in the images is not a strong one, since noise is by definition uncorrelated. As in all sliding window based filters, other type of defects such as lens aberrations cannot be corrected with this method, be with noise-free images or with noisy ones.

In the case of images retrieved from a camera, there will not be two identical images with random noise but, since two consecutive frames are usually very similar, they could be used instead of two identical ones. A similar technique has already been proposed in [1] for removing noise from video sequences, using two consecutive noisy frames to generate a cleaner frame.

This solution presents a drawback: both images must be as similar as possible in order for the filter to achieve a proper configuration. If that condition is not ensured, and the second frame presents several differences with the first one, as the system adapts to the task for which is trained, the resulting configuration will not only try to remove noise, but also cope with the differences by changing the input image. For instance, in the case of a moving real-time camera, two consecutive frames may have a small offset between them, so the filter will try to also move the input image in the same direction, and hence will behave very poorly if the movement is later done in the other direction. For that reason, it is important to perform a frame selection stage prior to evolution, to select the best pair of consecutive frames to train the system.

Hence, experimental results in the next section are shown in order to validate the equivalence of using noisy images for both input and reference images, and a detailed analysis of the assumption of using two consecutive frames as identical. Statistic results are shown in order to assess different frame selection algorithms in order to improve filter quality.

V. Experimental Results

The video sequence these experiments will use is the *Foreman* sequence, which has 300 frames, and has been resized to a size of 128×128 grayscale pixels. A 5% level of *salt and pepper* noise (randomly chosen pixels are changed to 0 or 255) is added to each of these frames in order to test if the filter is able to remove this noise. Quality measurements are done using SAE, measured against the original, noise-free images. The proposed system has been implemented on a Xilinx Virtex 5 LX110T FPGA, on an XUPV5 board.

A. *Evolution with noise-free reference versus noisy one*

In order to test the idea, three experiments have been run:

1. Traditional evolution, using a noisy image as input and the noise-free version of that image as reference.
2. Using as reference the same frame with noise (this is, both the input and reference image are the same frame with the same type of noise.
3. Using two consecutive noisy frames as both input and reference images.

All these experiments have been performed using the same evolutionary algorithm. The training process consists of choosing the best result among 5 independent evolutionary runs, each of them with 20 000 generations, 1 parent and 8 children per generation, using a mutation rate of K=3, that is, 3 elements (of a total of 25 elements are changed in every mutation operation). The total number of elements in the genotype, 25, corresponds to 16 PE slots + 8 input multiplexers + 1 output multiplexer. Every training sequence implies a total of 800 000 filter generations, reconfigurations and evaluations in order to get a single filter. Since the reconfiguration engine is capable of reconfiguring a PE in less than 100 us, and the evaluation of a candidate (filtering an image) is of the same order of magnitude, several thousands of circuits are generated and tested per second, so a complete training takes around 4 minutes, after which a filter will have been generated which will be able to filter images at a very high speed (200 Mpx per second), which makes it suited for real time applications.

After a filter is obtained, the whole sequence is filtered, and its output is compared, frame by frame, with the original noise-free sequence. This process is repeated with each of the 300 frames of the sequence, thus obtaining 300 different filters and so, 90 000 fitness (SAE) values for each of the three experiments. These values are shown in Fig. 4, next to the SAE values of the median filter, which is a filter commonly used for removing this type of noise [21], and the SAE the input frame had (the SAE of the non-filtered noisy sequence) are shown for comparison.

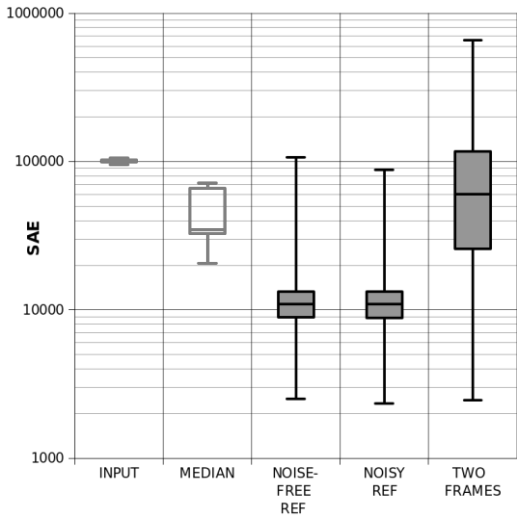

Fig. 4 SAE comparison between the noisy input, results of the median filter and the results obtained by the evolvable hardware system (lower is better).

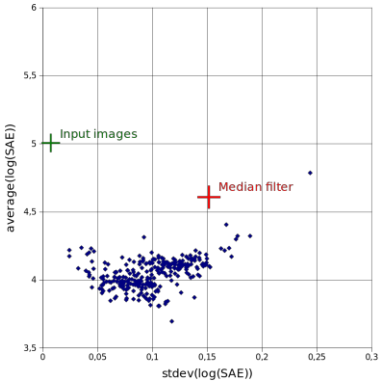

Fig. 5 Scatter plot of the results of the 300 different filters for the evolution based on a noise-free reference image. It shows the dispersion of the standard deviations and averages of log(SAE). Median filter and input image SAEs are also represented.

As can be seen, the filters obtained evolving with a clean reference are much better than the median filter, and most of them have less dispersion in the results, as it is shown in Fig. 5.

The results of evolving using two noisy versions of the same frame are almost identical to the ones using a clean reference, as can be seen in Fig. 4 and Fig. 6, even despite of not having a clean reference to train the filter with. Therefore, it is experimentally validated that noisy references can be perfectly used as a replacement for clean references for salt-and-pepper noise.

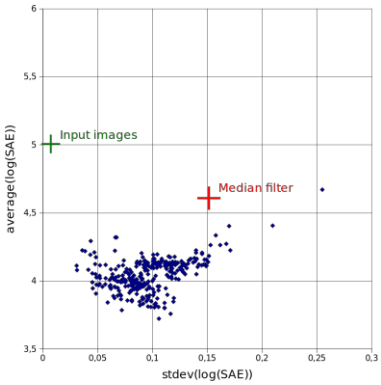

Fig. 6 Scatter plot of the filters obtained using non-correlated noisy images as reference.

Since two consecutive frames in a video sequence are expected to have little changes, one may think that the results of training a filter with two consecutive frames as input and reference would be similar to those of training it with the same frame. However, Fig. 4 and Fig. 7 show that this is not true: the results are worse and unstable in this case.

This is due to the video sequence having a lot of motion between frames, which causes the system to generate a filter that either tries to recreate that motion or attempts to generate an average image. Therefore, using any pair of arbitrary consecutive frames as both input and reference images for training is *not* a good solution, at least for video sequences with high amounts of motion or changes between frames. This problem will be dealt with in the next subsection.

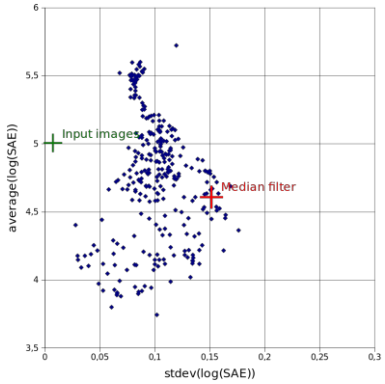

Fig. 7 Scatter plot of the different filters obtained by using two consecutive frames from a video sequence.

B. Consecutive frame selection method

A solution for the problem described in the previous subsection is to capture a sequence of frames, compare them in order to get the two most similar consecutive ones, and use them for training the filter. These comparisons can be performed on the noisy frames (that is, such as they are taken from the camera), and not necessarily with the noise-free ones, since the differences with and without noise are very well correlated, as it can be seen in Fig. 8.

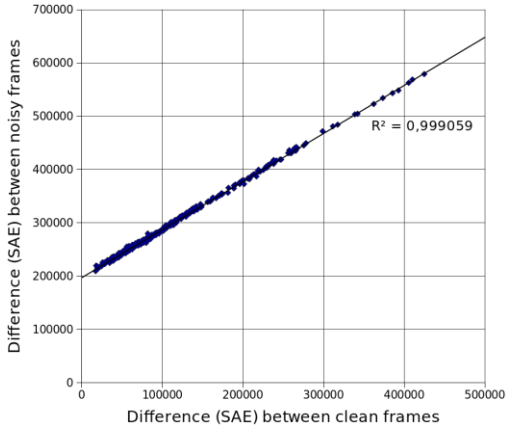

Fig. 8 Correlation between the values of the SAE between the clean consecutive frames, and the value of the SAE of the same frames with noise.

The computation of the difference between consecutive images can be done using the same hardware that is used for evolution, by configuring the filter as a pass-through filter that just copies the image and calculates the difference with the reference, so no additional hardware is needed for this.

The next question to solve is to see how long this sequence should be. In order to measure this, comparisons are performed on a subsequence of frames starting with each of the 300 frames on the whole sequence, and variable length (with rolling at the end of the sequence). As a result, 300 training processes are performed, but unlike with the previous experiment, some of these training processes will be performed with the same couple of images (local minimums around the whole sequence).

This experiment has been performed with a subsequence length of 10, 30, 60, 100 frames, and the whole sequence. As can be seen in Fig. 9 and Fig. 10, using a long enough frame sequence (of 30 to 60 frames), the obtained filters outperform the median filter in most cases, reaching performance values very similar to the ones obtained with the evolution with a clean reference when the sequence length is 100 to 300 frames.

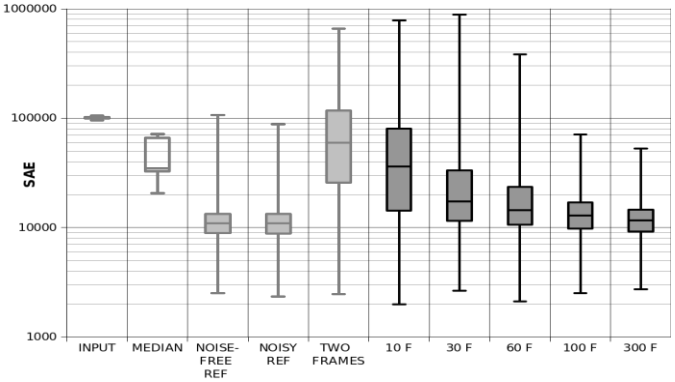

Fig. 9 Comparison of the previous results with the ones obtained by selecting the two most similar frames in a subsequence of 10, 30, 60, 100, and 300 frames respectively.

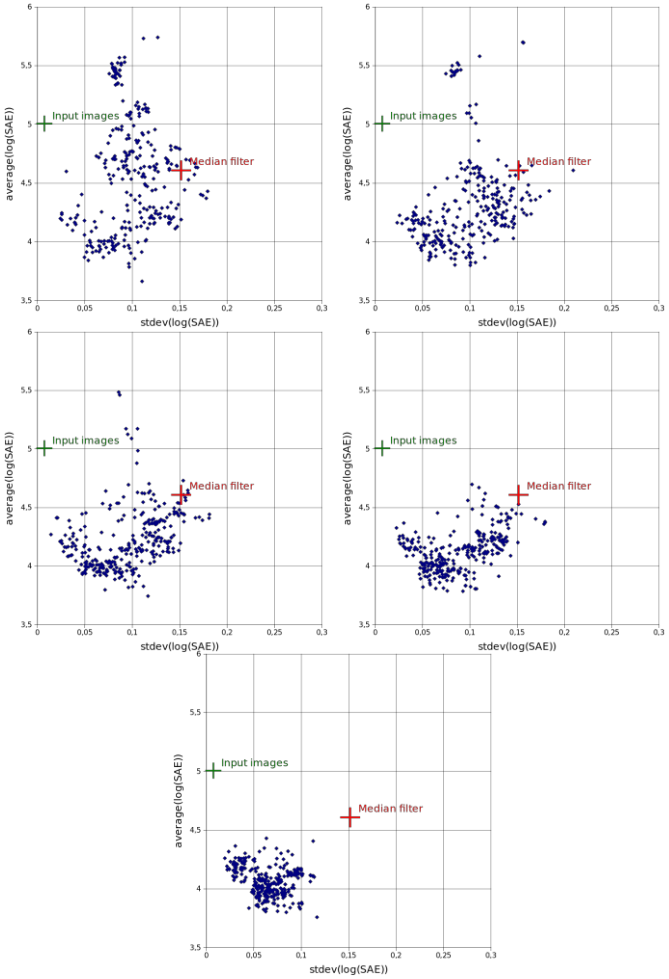

Fig. 10 Scatter plots of the different solutions: with 10, 30, 60, 100 and 300 consecutive frames respectively.

C. Experiments with other noise types and levels

As mentioned before, one of the advantages of evolvable systems is their generalizability. All the results shown in previous experiments have been obtained with the same type and level of noise (5% salt and pepper noise), but this system will adapt to other noise types and levels, as can be seen in Fig. 11. In each of these experiments, only 50 filters have been generated, each of them using a random starting frame and a subsequence length of 60 frames, assuming this is a sufficiently good method as derived from the previous subsection.

The types and levels of noise that have been used are 5%, 10% and 20% of salt and pepper noise (same as before), 5% impulse noise, in which randomly chosen pixels are replaced by a random value, or 5% burst noise, which simulates the effects of packet loss, which is seen as random horizontal white lines appearing on the image. The figure includes the results of the median filter (white boxes) next to the obtained filters (dark boxes). As can be seen, the results of the obtained filters are better than the ones of the median filter in most cases and similar for the 20% salt and pepper one. Nevertheless, it was shown in [22] that several cascaded arrays yield much better results.

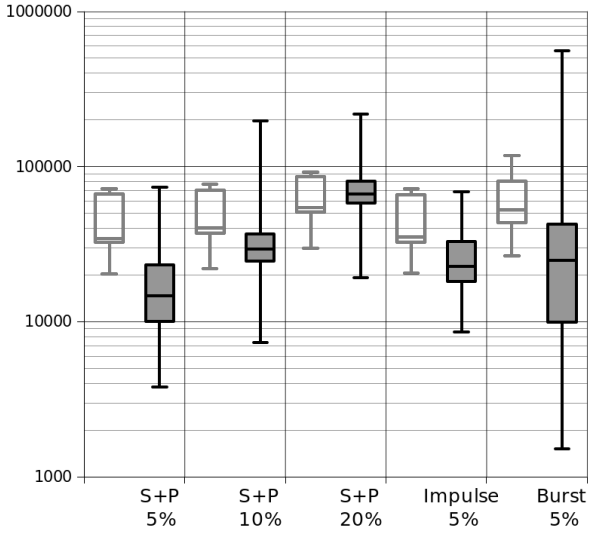

Fig. 11 Comparison among different types and levels of noise in the images. From left to right: salt and pepper noise with levels of 5, 10 and 20 %; impulse noise at 5%; burst noise at 5%.

Also some images are shown in Fig. 12 as an example of results obtained. This shows that the system is generalizable, which means it can be train with one image and be used with any other different images with the same type and level of noise, as the system adapts to the task for which is trained.

VI. Conclusions And Future Work

Evolvable hardware systems have shown to be a very good solution for solving adaptability problems such as environmental changes adaptation and fault tolerance. However, from a practical point of view, the need of having a reference to converge to is far from easy for fully autonomous systems. The proposed evolvable hardware system can successfully generate a configuration for filtering diverse types of noise, without needing a clean reference image nor a mechanism for classifying and emulating the noise. The noisy images used for training the filter can be either the same frame with non-correlated noise or two consecutive frames. In the case of two consecutive frames, a previous search must be done in order to find two frames that look alike as much as possible. With a system as proposed in [22], with dynamic scalability that allows to increase the number of arrays, this comparison task can be done in parallel, trying to find the best two consecutive frames while the filtering task is kept.

This system relies on the randomness of noise, which causes noisy pixels to be placed on different positions of both images. As a result, this system will not preform very well with noise with other causes such as damaged pixels on fixed positions (e.g. dust on the camera lens), or noise that affects all pixels of an image rather than a few ones (for instance the amplifier noise, which adds an uncorrelated (white) error to all pixels on the image, so there will not be any clean pixel on the reference image that can be used as actual reference).

VII. References

[1] Zhou, Xiang; William G. Wee.; "Adaptive order statistic filters for noise characterization and suppression without a noise-free reference." Communications, 1998. ICC 98. Conference Record. 1998 IEEE International Conference on. Vol. 3. IEEE, 1998.

[2] Otero, A.; Salvador, R.; Mora, J.; de la Torre, E.; Riesgo, T.; Sekanina, L.; "A fast Reconfigurable 2D HW core architecture on FPGAs for evolvable Self-Adaptive Systems," 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS).

[3] Salvador, R.; Otero, A.; Mora, J.; de la Torre, E.; Sekanina, L.; Riesgo, T.; "Fault Tolerance Analysis and Self-Healing Strategy of Autonomous, Evolvable Hardware Systems," International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2011.

[4] Bovik, A. C. "Handbook of image and video processing," Academic Press, (2010).

[5] Vijaykumar, V. R.; Ebenezer, D.; Vanathi, P. T.; "Detail preserving median based filter for impulse noise removal in digital images," In Signal Processing, 2008. ICSP 2008. 9th International Conference on (pp. 793-796). IEEE 2008, October.

[6] Hwang, H.; Haddad, R. A.; "Adaptive median filters: new algorithms and results," Image Processing, IEEE Transactions on, 4(4), 499-502. 1995.

[7] Wang, Z.; Zhang, D.; "Progressive switching median filter for the removal of impulse noise from highly corrupted images," Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, 46(1), 78-80. 1999.

[8] Marshall, S., "New direct design method for weighted order statistic filters," Vision, Image and Signal Processing, IEE Proceedings - , vol.151, no.1, pp.1,8, 5 Feb. 2004

[9] Chan, R. H.; Ho, C. W.; Nikolova, M.; "Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization," Image Processing, IEEE Transactions on, 14(10), pp 1479-1485, 2005.

[10] Aiswarya, K.; Jayaraj, V.; Ebenezer, D.; "A new and efficient algorithm for the removal of high density salt and pepper noise in images and videos," In Computer Modeling and Simulation, 2010. ICCMS'10. Second International Conference on (Vol. 4, pp. 409-413). IEEE.

[11] Vasicek, Z.; Sekanina, L., "Novel Hardware Implementation of Adaptive Median Filters," Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on , vol., no., pp.1,6, 16-18 April 2008

[12] Fahmy, S.A.; Cheung, P. Y K; Luk, W., "Novel FPGA-based implementation of median and weighted median filters for image processing," Field Programmable Logic and Applications, 2005. International Conference on , vol., no., pp.142,147, 24-26 Aug. 2005

[13] Caban, D.; "FPGA implementation of positional filters," In Design of Embedded Control Systems (pp. 243-249). Springer US. 2005

[14] L. Sekanina, "Virtual Reconfigurable Circuits For Real-World Applications Of Evolvable Hardware" Proc. of the 5th international Conf. on Evolvable systems: from biology to hardware. ICES 2003, vol. 2606, pp. 186-197.

[15] Vasicek, Z.; Sekanina, L.; Bidlo, M., "A method for design of impulse bursts noise filters optimized for FPGA implementations," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010* , vol., no., pp.1731,1736, 8-12 March 2010

[16] Miller, J. F.; "An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach," *Proceedings of the Genetic and Evolutionary Computation Conference* (Vol. 2, pp. 1135-1142), July 1999

[17] Krishna, K. S. R.; Reddy, A. G.; Prasad, M. G.; Rao, K. C.; Madhavi, M.; "Genetic algorithm processor for image noise filtering using evolvable hardware," *International Journal of Image Processing*, *4*(3), 240-250, 2010

[18] R. Salvador, A. Otero, J. Mora, E. de la Torre, T. Riesgo, and L. Sekanina, "Evolvable 2D computing matrix model for intrinsic evolution in commercial FPGAs with native reconfiguration support" Proc. of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems, IEEE Computer Society, 2011, pp. 184-191.

[19] Torresen, J.; Senland, G.A.; Glette, K., "Partial Reconfiguration Applied in an On-line Evolvable Pattern Recognition System," *NORCHIP, 2008.* , vol., no., pp.61,64, 16-17 Nov. 2008

[20] Glette, K.; Torresen, J.; Hovin, M., "Intermediate Level FPGA Reconfiguration for an Online EHW Pattern Recognition System," *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on* , vol., no., pp.19,26, July 29 2009-Aug. 1 2009

[21] G.R. Arce, "Nonlinear Signal Processing: A Statistical Approach", Wiley:New Jersey, USA, 2005.

[22] Gallego, Á.; Mora, J.; Otero, A.; de la Torre, E.; Riesgo, T.; Salvador, R; "A Novel FPGA-based Evolvable Hardware System based on Multiple Processing Arrays", Reconfigurable Architectures Workshop, Parallel & Distributed Processing, 2013. IPDPS 2013. IEEE International Symposium on.

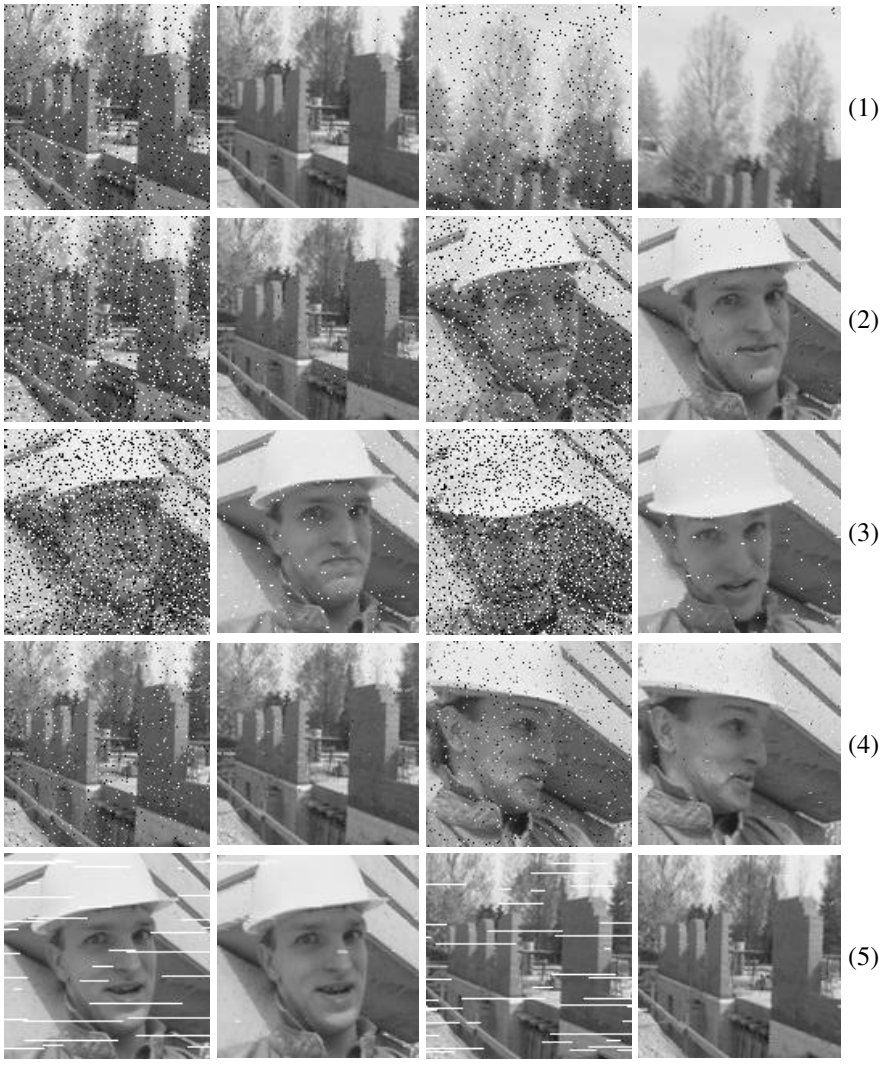

Fig. 12 Some of the obtained results. (a) Input frame used for training. (b) Result of filtering such frame. (c) Another frame in the sequence. (d) Result of filtering frame in (c) with the filter trained with (a). Rows show 5%, 10% and 20% salt and pepper noise, 5% impulse noise, and 5% burst noise.